Classification using Neural Networks – Witchcraft or Wizardry?

A Comparative Study of the Application of Classification Neural Networks that Exploit the Use of Minority Class Boosting - Synthetic Minority Over-Sampling Technique (SMOTE) and Restricted Boltzmann Machines (RBMs)

Daniel Dixey and James Muller, City University London, INM427 Neural Computing

Motivation—Neural computing methods can be applied to a myriad of applications. These include classification, clustering, pattern recognition and associative memory. Their approach stands in stark contrast to that of more mainstream machine learning techniques where a panoply of 'competing' algorithms (statistical and non-statistical) can be used to varying degrees to address a problem. The implementation and evaluation of methods conducted as part of this report aims to show the efficacy of neural network methods in this domain and their potential for outperformance in classification exercises over other algorithms. If it were not for the sheer scale of data streams today, time constraints and limited human resource availability, 'experts' (often called mechanical turk's) performing 'human intelligence tasks' (HITs) such as 'case labelling' could arguably do a better job (notwithstanding their own inherent fallibility). Instead, these 'experts' focus on interpretation of the exceptions within a dataset.

Index Terms—Neural Computing, SMOTE, restricted boltzmann machines, bank marketing classification, backpropagation, python

1. Introduction

1.1 Description of problem

Mis-classifying a dataset can be argued to promote inefficient and ineffective distribution of resources, whether in a private (corporate) domain or on a wider public level. Misinterpreting the outcome of a set of stimuli provided by the details of a case can also have extremely serious 'local' consequences (for example, wrong diagnosis of a serious illness). The limitations on human agents as a solution to this problem argue for an alternative solution, especially in a world of wildly accelerating data volumes. 'Machine' solutions (and neural networks in particular) are one such approach.

1.2 Aim

The aim of this paper is to explore the benefits of neural network techniques in the investigation, characterization and classification of bank marketing campaigns. The intention is to provide evidence concerning the application of neural network classification methods in this domain. It is also intended to test the benefits of oversampling techniques in increasing the accuracy of classification.

1.3 Research Question

"What evidence can we provide for the use of neural networks (and oversampling) to a classification problem (prediction of take-up of bank marketing campaigns)?"

1.4 Scope of study

The scope of the paper is limited to the confines of the dataset (a Portuguese bank marketing campaign [1], [2]). 'Novelty' is provided by our exploration, (pre-)processing and transformation of the dataset, implementations used, scripting, choice of algorithms/dataset pair, initial configuration/set up of parameters, combination of and sensitivity to parameters searched under cross–validation and grid search, subjective choice and rationale for the 'preferred best' model, testing, analysis and critical evaluation and comparison against an example set of machine learning algorithms applied to the same task. In essence, 'uniqueness' is provided in the holistic approach we have taken. Areas for further work are discussed at the end of the paper. The techniques will set a non-boosted multi-layer perceptron (with back propagation) against two examples of boosted (oversampled) multi-layer perceptron's using Synthetic Minority Oversampling Techniques (SMOTE) [3] and Restricted Boltzmann Machines (RBM).

1.5 Objectives

Specific targets will support achievement of the aims of the project. The key objectives are:

- Implementation of neural networks on a bank marketing campaign dataset for classification purposes
- Determination of an 'optimum' neural network model under various training and validation conditions
- Investigate the generalization error/'over fitting' and reliability of the chosen models on a test dataset
- Uncover the parameters that 'make the most difference' to the result through a grid search
- Conduct a baseline comparison of the chosen neural network algorithms vs. a group of standard machine learning algorithms and a non-boosted multi-layer perceptron
- Provision of evidence to support neural network application to this type of classification problem
- Interpretation of the balance drawn between performance and complexity
- Provide a basis for further study of the 'zoo' of parameters/settings that could be brought to bear

2 Approach

2.1 Data

2.1.1 Dataset

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

2.1.2 Population v sample

The dataset used is described as very close to that in the research that initially used it. Our analysis and report assumes that our dataset is representative of the full population. If this assumption does not hold the results will be subject to review. It is also assumed that the full population is indicative of such datasets for the purposes of generalization and so subject to the same caveat.

2.1.3 Data Properties and Processing

The table below describes the data structure, type, (pre-) processing and transformation steps applied. An iterative 3-stage procedure was adopted to 'clean'/scrub the data, apply visual exploration measures and decide/implement data transformative (encoding) techniques to make the data ready for analysis.

Input (Attribute) Information					
Data Size: 41,184 instances (reduced from 41,188)					
Data Structure:					
Pre-transformation - 21 attributes (20 plus class variable – 11 categorical, 10 numeric)					
Post-transformation – 63 attributes (62 plus class variable – 63 numeric)					
		Transformation		(Pre-)Processing	
Grouping	Type	Yes/No	Action	Yes/No	Action
· Client Data (7)	Categorical (10)	Yes	Binarization [0 or 1] and separation into different columns	Yes	Provenance of dataset retained as far as possible. Removal or replacement of instances or attributes limited to: 4 instances removed where duration of call recorded as nil. See comment below.
· Last Contact (4)					
· Other (4)					
· Social/Economic (5)	Numeric (10)	Yes	'Normalizing' - rescaled to {0,1}		
· Class (1)	Class variable (1) (y/n)	Yes	Binarization [0 or 1]		

2.1.4 Recombination of Columns

The decision was taken not to recombine the increased number of attributes created by binarization and separation of each of the discrete outputs from the categorical variables. This increases the dimensionality of the inputs to the classification and sparsity of some of the attributes created.

2.1.5 Attributes 11 and 13

The dataset contains one attribute (no. 11) that represents the duration of the last contact with a potential customer. Since a value of nil indicates a 'no' the classification task will in theory be influenced by the weight attached to the predictive power of this variable to the detriment of the predictive power of the other attributes. We have chosen to retain dataset provenance where at all possible but have removed these instances.

Attribute 13 contained entries relating null inputs to a numerical value. We normalized the data to 'rescale away' the magnitude deviation otherwise introduced.

2.1.6 Class imbalance

The dataset shows that the class variable is weighted heavily towards one class ('not taken up'). In order to train the neural network algorithms efficiently i.e. with enough variety in the data to enable generalization we will introduce a class rebalancing element (via SMOTE and RBM) [4]. This will aim to provide an approximate doubling in the smaller class (to c.20% of the total) and enhanced performance of the 'best performing' models when applied to the test dataset.

2.1.7 Basic statistics

Some exploratory basic statistics were generated for the numerical attributes to give a sense of the dynamics at play in the dataset. Data provenance was retained subsequent to their generation.

Measure	age	duration	campaign	pdays	previous	emp.var.rate
count	41184	41184	41184	41184	41184	41184
mean	40.0235285548	258.3100961538	2.5673805361	962.4719065657	0.172979798	0.0819201632
std	10.4210443004	259.2793440056	2.7698947108	186.9196375829	0.4949221777	1.5709598537
min	17	1	1	0	0	-3.4
25%	32	102	1	999	0	-1.8
50%	38	180	2	999	0	1.1
75%	47	319	3	999	0	1.4
max	98	4918	56	999	7	1.4

2.1.8 Comment on dimensionality reduction

The analysis will be based on the (expanded) attribute set. Dimensionality techniques aimed at reducing the complexity of the classification task could be a subject for future work should the results prove inconclusive or overly expensive in production [5]. Such techniques could include Principal Components Analysis (PCA), Multi-Dimensional Scaling (MDS), Self-Organizing Maps (SOMs) or RBMs. Other dimensionality approaches such as using an auto-encoder are reserved for tasks requiring the reproduction of inputs as outputs (such as translation). Under such methods sparse coding uses a small subset of neurons to hold key patterns that can then be used to reproduce the original input as an output.

2.2 Neural network models chosen

2.2.1 Multi-layer perceptron with back-propagation (MLP BP) no boost (i.e. no oversampling)

MLP BPs are directed, feed forward, fully connected, acyclic neural networks that can be applied to learning 'tasks'. They consist of an input layer, output layer and one or more hidden layers of nodes (neurons). No lateral connections exist between neurons and no feedback is permitted. They are considered biologically plausible representations of the workings within a human brain. MLP BPs are initially unsupervised (initialized) and subsequently fed labelled data to help learning. They are an example of semi-supervised/reinforcement learning algorithms.

2.2.2 Description of training (learning) and testing

Steps for training a MLP BP are as follows:

- Initialize network (configuration of parameters)
- Conduct a feed forward pass (propagate a signal through the network) to compute activations at all (hidden) layers and then at an output layer to produce an output
- Measure the difference (reconstruction error) between the output obtained and target required
- Back propagate the error through the neural network and perform weight updates (requires differentiable activation functions and (typically) uses the chain rule to iteratively calculate gradient descent optimization i.e. a general case of the delta function)
- Repeat for a series of training cycles until the error reaches a (small) target and test for generalization

2.2.3 Model 1 – MLP BP plus boost (SMOTE) (oversampling to boost minority class)

SMOTE oversampling is introduced to the dataset for subsequent investigation. The dataset is subject to class imbalance (88.735% 'no':11.265% 'yes'). SMOTE uses the k-means clustering algorithm to provide extra 'artificial' samples of the imbalanced class labelled instances. This is achieved by taking each minority class sample and introducing synthetic examples along the line segments joining any of the k minority class nearest neighbours. Steps in the process are as follows:

- Iterate through multiple variations of k and N (percentage of new synthetic samples):
  - Identify k – no. nearest neighbours (in our case k=5) and n– amount of SMOTE to use (in our case n=100%)
  - Fix the inputs on the neural network: learning rate = 0.1; number of neurons = 80; number of epochs = 80,000
  - Benchmark the results and establish a clear 'winner'

It is important to be aware K-means clustering can perform poorly in the face of high dimension data. Other techniques (DBScan, hierarchical clustering) avoid some of the particular problems faced.

2.2.4 Model 2 – MLP BP plus Boost (RBM)

An RBM is a stochastic, recurrent neural network that learns a probability distribution over its inputs for various tasks, including generating samples. RBMs contain visible and hidden nodes, can have symmetry between each set of nodes but has no connections between nodes (i.e. they are restricted and so conditionally independent).

2.2.5 Training an RBM to provide samples

RBMs learn (global) energy function minima by finding thermal equilibria. The visible units are 'clamped' (receive information from the environment i.e. the existing class-imbalanced samples). The network is then trained so it matches the 'real' probability distribution over the hidden/visible units (the positive phase) with that of a freely running network (the negative phase). The process recreates the input data (a 'fantasy', including our new samples). By 'matching' probability distributions, classification (or reproduction of similar samples) can be achieved. RBM learning [6] typically uses gradient descent and Gibb's sampling (or Markov chain Monte Carlo methods) ('Product of Experts') or combination methods (latent variables feed further layers for feature creation) such as Contrastive Divergence.

In both models the oversampling approximately doubled the imbalanced class (adding an extra c.4, 000 instances). To guarantee that the number of 'fantasies' generated by the RBM was identical was proportional the decision was made to set up the number of hidden neurons to this value. The RMB machine was then test with a variety of parameters settings to ensure potential optimality in finding a good approximation to the joint probability distributions at each neuron. For continuity the learning process that was used for the RBM was implemented with Contrastive Divergence with a stochastic learning process, where one random per epoch is used to adjust the weights.

Neurons (Hidden Layer)	c.4000	
Learning Rate	0.0075	
Number of Epochs	400	
Initiations of Weights	Range	[-0.25, 0.25]
	Distribution	Uniform

2.3 Advantages and disadvantages of MLP BP with Boost (SMOTE or RBM)

2.3.1 Advantages

- Popular and widely used
- 'Ground truth' standard and biological plausibility (certain configurations)
- Robust and adaptable
- Can approximate any continuous function (Universal Approximation Theorem [7])
- Neural networks have the capacity to detect complex (nonlinear) relationships between dependent and independent variables that traditional machine learning algorithms can struggle with (examples include deep learning, convolutional neural networks and stacked RBMs)
- Boosted neural networks can 'learn' the unbalanced class more efficiently than if un-boosted

2.3.2 Advantages

- Simplicity/lack of biological plausibility (certain configurations, including choice of activation function)
- Initialization can greatly affect performance
- Determination of 'optimum' parameter set
- Potential for convergence to local minima/'high' error states (vanishing gradients)
- Convergence to (local or global) minima can be slow
- Susceptible to 'over fitting' if capacity (number of neurons) allows sufficient learning
- Randomness inherent in procedure can prevent replication of results
- "Black box" nature limits understanding of processes at work
- Computationally demanding (even with gains in e.g. parallel processing and use of GPUs)

2.4 Second order methods

We have extended the scripting in a number of ways. This has meant making choices about the limits to the dimensions of our analysis. Second order methods that take into account curvature of the error surface (e.g. Hessian matrices or Levenberg-Marquardt methods) have not been introduced and are an area for future work.

2.5 Outline methods/algorithm

The high level 'pseudo' processes for the competing neural networks can be written as follows:

- Import dataset
- Pre-process dataset
- Transform and combine dataset into accessible format
- Rebalance class variables (SMOTE and RBM)
- Create training/validation and test datasets
- Define configuration and set-up elements
- Determine parameter ranges
- Train models
- Test models
- Return all results, analyze, evaluate and interpret

2.6 Script development

Key elements of script development are as follows:

- Syntax - Python code, making use of additional modules namely Scikit Learn, Pandas, Numpy and Scipy
- Commenting - Code heavily commented to assist understanding and for validation by readers and colleagues
- Choice and Difficulty of Methods – functions developed from existing (free) function modules, including user-defined functions to wrap and combine functionality together. Implementation of other author code and pseudo-code required extensive, multi-stage development and iterative brainstorming

2.7 Hypothesis

2.7.1 Hypothesis statement

H0: The null hypothesis states that neural networks cannot outperform 'white box' machine algorithms
The nature of 'black box' algorithms such as neural networks that prevent easy interpretability of the methods by which tasks (like classification) are performed mean that more accessible traditional machine learning methods should be preferred. "Ockham's razor" should apply.

H1: The alternative hypothesis states that neural network dynamics provide performance results that demand their continued implementation and development as a viable alternative to more traditional machine learning algorithms

2.8 Assumptions and settings

A number of assumptions were made concerning the choices to be made. These are as follows:

- Stochastic sequential learning (offline)
- Uniform distribution in training/validation folds or held-back test data/stratified sampling applied
- Epochs per training fold
- Activation function, cost function, initialization of weights
- Number of hidden layers and nodes per layer
- Learning rates
- Grid search technique (without extended sensitivity analysis or manual, scenario based approaches)

2.9 Limitations

The methods, results and evaluation were restricted in a number of ways as follows:

- Scripting and Algorithms - type of complexity
- Configuration and Parameters Used – the choices made e.g. no momentum, regularization, second order methods; no changes between layers
- Cross-validation methods adopted

2.10 Restrictions

The restrictions on wider applicability/generalization of the results and findings are as follows:

- Scope and quality of dataset
- 'Black box' nature of nodes (unable to describe workings)
- Inability to infer causation from results
- Interpretability of results (what drives them)
- Stability and behaviour of neural network models (randomness inherent)

3. Training and evaluation methodology [8], [9], [10], [11], [12], [13]

3.1 Design

3.1.1 Split of dataset

The dataset was split in the following way:

- Sampling: The test set was separated from the rest of the dataset that will be used for training and validation. A stratified sampling procedure was adopted to ensure that the training/test sets were of comparable nature.
- The synthetic samples were added to the training data.

3.1.2 Training and validation

Training and validation were conducted on the base case and two competing algorithms using ten-fold cross-validation. Under this procedure, the training dataset was split into ten folds. For each of ten runs, a single 10% validation fold of the training dataset (each entirely different from any of the other validation folds) was reserved for validation and the remaining 90% applied each time for training.

Ten–fold cross validation was run each time on a combination of 576 parameter settings (models), each comprising a different combination of parameters. In total, 5,760 training and validation runs were conducted on each algorithm. This returned 10 results per parameter combination (model). These were 'averaged' and all 576 model results returned for each algorithm.

Following the cross-validation procedure, the 'preferred' 5 models were chosen for each of the three alternatives.

3.1.3 Testing

The unseen test dataset was processed to remove the class variable. The test dataset was then run on the 5 'winner' models for each of the algorithms, accuracy and supporting statistics recorded and extent of generalization error noted.

3.1.4 Choice of parameters

Item		MLP BP No Boost	MLP BP plus Boost (SMOTE)	MLP BP plus Boost (RBM)
Class rebalanced		No	Yes	Yes
Batch v. Sequential		Stochastic sequential – 1 random sample per epoch		
No. of Hidden Layers		1	1	1
Activation Function	Hidden/ Output	Sigmoid (Tanh)	Sigmoid (Tanh)	Sigmoid (Tanh)
Cost Function		Cross entropy	Cross entropy	Cross entropy
Initial Weights	Start	[-0.25,0.25]	[-0.25,0.25]	[-0.25,0.25]
	Distribution	Uniform	Uniform	Uniform

Note: When calculating classification statistics the prediction vector was converted using a piecewise constant (step) function using: where $x < 0.5 \rightarrow 0$ and where $x >= 0.5 \rightarrow 1$. Bias = 1.

3.1.5 'Stop criteria' v 'early stop'

Our analyses varied the number of passes (epochs) to constrain each run of a model.

Note: 'Stop criteria' are a form of regularization (see below). Alternative stop criteria include a minimum accuracy level (and no change for a number of passes), or a minimum threshold of runs falling below a certain error level. 'Early stop' methods include halting methods when the validation error begins to rise.

3.1.6 Processing method

The analysis was conducted on the basis of stochastic sequential processing. Sequential alternatives make the best use of resources where data is at scale and arriving in real time or items are fed in one by one.

3.1.7 Activation function

We used a variation of the tan sigmoid (tanh) function in our analysis at all stages where tanh x = sinh x/cosh x = (1-e^-2x)/(1+e^+2x). As recommended in the text by Yann Lecun [14] the activation function has been used in the script, the actual activation function used is shown below. Hyperbolic Tan functions can be used to approximate almost any function and can map numbers to a [0,1] range. They reduce the chance of saturation in hidden layers and are usefully symmetrical around zero (logistic functions are not). Since the integral of any "bump-shaped" function is 'S-shaped' and we may rely on a normally distributed error function the choice is doubly useful. Many processes exhibit a process that's starts slowly, accelerates and then approaches a limit over time. Alternatives that could have been adapted for use include sigmoid, logistic and softmax functions.

$$\text{Activation Function: } f(x) = 1.7159 \tanh\left(\frac{2x}{3}\right)$$

$$\text{Derivative : } f'(x) = 1.14393(1 - \tanh^2\left(\frac{2x}{3}\right))$$

3.1.8 Cost (error) function

Cross entropy is often the recommended cost function for classification and especially for imbalanced data (as in our dataset). It penalizes mis-classification heavily by faster learning. Cross-entropy outperforms quadratic and similar cost functions by learning quicker in the face of 'completely wrong' answers. This can come about easily in neural networks, either by incorrect initialization of bias/weights or using smooth activation functions that predict incorrectly according to a probability distribution at the output layer. It is usual for the aforementioned reasons to use smooth activations and cross entropy in tandem when addressing the make-up of multi-layer perceptrons.

3.1.9 Initialization of weights

The starting weights are a critical element in the success or otherwise of a neural network. They contribute to the starting point on the error surface, likelihood of reaching local (vs. global minima) and speed of learning. We pick different random values other than 0 or 1. This breaks symmetry (same signal at each node) and helps avoid 'greedy' algorithms finding non-optimal minima. Varying weights over many random combinations may be a useful area for future research.

3.1.10 Number of hidden layers

According to the Universal Approximation Theorem virtually any continuous function can be approximated and represented by a single hidden layer structure for a multi-layer perceptron (although not the function's learnability). We adopt this principal and vary the number of nodes (neurons) within this layer for the purposes of the analysis and evaluation. Since back propagation for multiple hidden layers quickly shows limited returns the predictive power of a multi-layer perceptron can be well-harvested using a single hidden layer.

3.1.11 Number of hidden layers

Bias allows the output of each neuron to be sensitive at values different from zero. This is especially useful for classification (0 or 1 outputs).

3.1.12 Parameters varied

A series of 576 combinations of parameters were derived from the following parameter ranges:

Parameter Varied	Range/Selected Values
Epochs (passes) – for training/validation/testing	[60000,120000], step size of 10000
Number of nodes (neurons) in hidden layer	[60,120], step size of 5 neurons
Learning Rate	[0.05,0.12], step size of 0.01

Note: Each epoch represents one sample from the dataset. The learning rate subtracts a ratio of the gradient of the error function from the weight to help generalization. Certain other parameters were not used. Examples follow: 1) Momentum is an additional learning factor applied to weights to try and avoid local minima. 2) Regularization ensures smoothing of learned models to prevent 'over fitting'.

4 Results

4.1 Training and validation

The 'Top 5' models from the grid search training and validation exercise were chosen based on accuracy as follows:

Model Rank	Acc (%)	F1	CE	Nodes	LR	Epochs
MLP BP No Boost						
1	90.60	0.44	3.25	65	0.11	100,000
2	90.54	0.44	3.27	90	0.08	110,000
3	90.53	0.45	3.27	65	0.10	90,000
4	90.51	0.45	3.28	80	0.08	110,000
5	90.51	0.43	3.28	90	0.06	100,000
MLP BP plus Boost (SMOTE)						
1	91.44	0.76	2.96	80	0.07	100,000
2	91.38	0.76	2.98	70	0.09	110,000
3	91.38	0.75	2.98	75	0.05	110,000
4	91.37	0.76	2.98	70	0.12	110,000
5	91.37	0.76	2.98	60	0.05	100,000
MLP BP plus Boost (RBM)						
1	91.35	0.76	2.99	80	0.10	90,000
2	91.34	0.75	2.99	95	0.10	110,000
3	91.33	0.75	3.00	85	0.11	110,000
4	91.30	0.75	3.00	80	0.05	110,000
5	91.30	0.75	3.00	65	0.07	100,000

Note: Accuracy, F1 and Cross Entropy figures are averages.

4.2 Testing

Each of the 'Top 5' models for each algorithm was then tested using the test dataset. The 'Top 3' preferred models (one from each category of algorithm) are shown in green. The determining statistic was chosen to be NPV (see discussion later). The remainder of the statistics returned support this choice. The results recorded were as follows:

Model Number (Training Rank)	Acc (%)	F1	CE	ROC_AUC	NPV
MLP BP No Boost					
1a	89.39	0.17	3.67	0.55	9.48
2a	89.10	0.12	3.77	0.53	6.47
3a	90.51	0.40	3.28	0.64	28.66
4a	90.26	0.35	3.36	0.61	23.49
5a	89.56	0.59	3.61	0.80	67.03
MLP BP plus Boost (SMOTE)					
1b	90.58	0.53	3.25	0.72	47.63
2b	90.53	0.43	3.27	0.65	32.90
3b	89.90	0.30	3.49	0.59	19.18
4b	90.58	0.50	3.25	0.69	42.03
5b	90.29	0.46	3.35	0.67	36.21
MLP BP plus Boost (RBM)					
1c	90.55	0.51	3.26	0.70	43.75
2c	90.65	0.52	3.23	0.70	44.40
3c	90.65	0.53	3.23	0.71	46.77
4c	90.36	0.38	3.33	0.62	26.08
5c	90.21	0.39	3.38	0.63	28.02

Note: ROC_AUC is the area under the curve of an Receiver Operating Characteristics plot (True Positives v False Positives. NPV = Negative predicted rate = TN/(TN+FN).

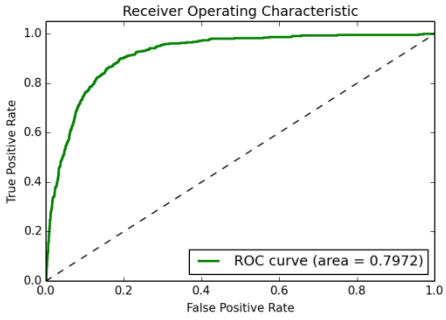

Figure 1: Illustration of the receiver operating curve plot of model 5a

4.3 Comparison with machine learning models

For a fuller comparison of the extent and robustness of our neural network algorithms the analysis ran a training/validation and testing exercise on a machine learning model set. Each machine learning model was subjected to the following steps in training/validation (no class boost, ten-fold cross-validation, default configuration, set-up and parameter set) and then tested on the test data. The results are contained in the table on the right and in figure 3:

Rank	Model	Accuracy (%) (Test)	NPV
1	MLP BP No Boost	89.56	67.03
2	Gradient Boosting Classifier	91.57	51.51
3	MLP BP plus Boost (SMOTE)	90.58	47.63
4	MLP BP plus Boost (RBM)	90.65	46.77
5	Random Forests	90.99	42.46
6	Support Vector Machines	91.33	41.38
7	AdaBoost Classifier	91.26	41.16
8	Logistic Regression	90.17	21.55
9	K-nearest neighbours	89.66	0.00

5 Analysis and evaluation of results

5.1 Performance

5.1.1 Comparison of results

Our results suggested that, for the methods and setting used:

- MLP BP No Boost performed best (by quite some margin) on our chosen key statistic of NPV (domain specific)
- Neural network models performed comparably well to machine learning models on accuracy (3 of top 4 places)
- Boosted models performed better than un boosted models on both accuracy measures
- SMOTE and RBM oversampled models performed very similarly across the board
- Gradient boosting classifier outperformed the other machine learners
- The worst performers were machine learners logistic regression and k-nearest neighbours
- Neural networks ran best on 80-90 nodes (single hidden layer), a learning rate c. 0.1 and c.100,000 epochs

5.1.2 Importance of domain knowledge

Expertise in neural network dynamics and experience of their implementation and performance would be critical in a further review of performance. We used parameters that appeared sensible in the circumstances and confines of the remit of the study and attempted to avoid values and ranges known to cause problems for neural networks. This is especially the case of the oversampling methods. We could have a priori expected training on extra positive instances to decrease the number of false negatives.

5.1.3 Importance of 'cost matrix'

The performance achieved must be equalized according to the costs of each output state (confusion matrix). The preferred model depends heavily on the future attachment of costs to, in particular, false negatives.

5.2 Complexity

5.2.1 Data

Our dataset was relatively clean and ideal for the classification task. We did not apply dimensionality reduction which may have affected both efficiency of the techniques used and learning power of the models chosen.

5.2.2 Models and methods

The approach used standard neural network set-up with just a single layer. Oversampling methods added sophistication and was the basis for our comparison. We used 'gold standard' ten-fold cross-validation and exhaustive grid search procedures which extended the scripting challenges. The many settings required careful collection, collation and handling/review (statistics generated).

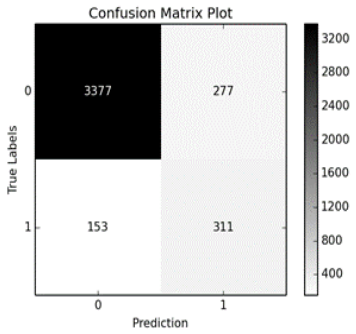

Figure 2: Confusion matrix plot of model (5a)

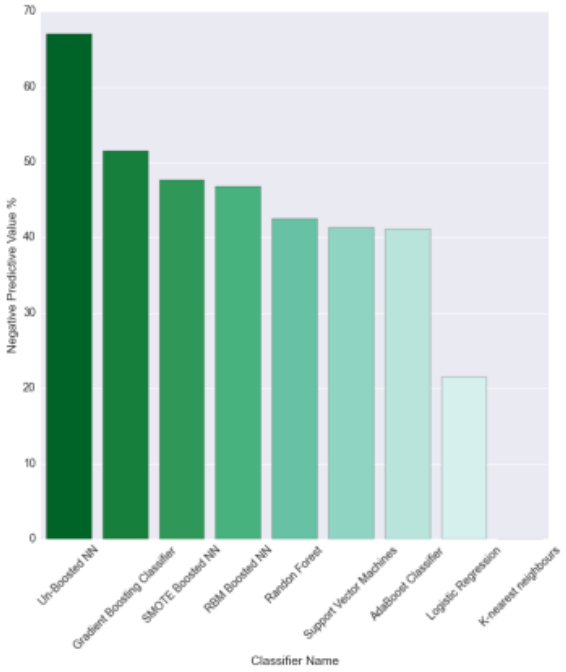

Figure 3: Negative Predictive value results for all NN and ML models tested

5.2.3 Configuration and parameters

We examined many initial parameter settings and configuration profiles. The real complexity is in management of the process and iteratively adjusting the models as they run over time.

5.2.4 Demands on Resources

The effort/complexity required to achieve the results can be partly gauged by the following operational measures:

Computational Dimensions	Measure and Value
Time to Process (High-Medium-Low)	High (Total Time Processing Python Script 29 hours)
Average Memory Use across Computations	42%
No. of System Failures during Implementation	2 (due to script errors)

Note: All scripting estimates recorded manually. Time measured via internal 'clock'.

The fullness of our approach created demands on resource management. These were handled within the time allocated and the results and report generated.

`

5.3 Performance v complexity

We ideally look for a high performance (minimum cost) algorithm that is complex enough to capture the granularity required but simple enough to make the analysis tractable (doable) within the constraints set.

In the case of bank marketing programmes, the fixed cost nature of operational sites ('call centers') renders the minimization of false negatives critical. Every call made to a contact who is in fact likely to 'take a product' must be harvested to provide a profit contribution with which to pay down a fixed cost platform. The incremental cost of false positives (calling a 'cold' contact via the 'dialer') is often low.

6 Conclusions

6.1 Findings v objectives set

Our research concludes that a solid implementation of neural networks was achieved with good performance on absolute and relative grounds and parameter ranges determined. An optimal model was found (no boost). Indeed, the ROC achieved matched that in the reference papers for the Portuguese bank marketing exercise. The work achieved this performance within manageable complexity.

6.2 Lessons learned

We learnt a number of lessons from the analysis. The dataset must be intensively studied at the outset and critical decisions made on its determination. This helps drive scripting and set-up of the experiments. Methods can quickly become convoluted and slow. It can also sometimes be hard to interpret all the stages being undertaken. Training and testing also requires careful planning to find a 'middle ground' that applies rigor and simplicity in equal measure. The final 'choice' of winner can also be extremely subjective. Expert knowledge and sheer brute force often rank equal in importance. Heuristics to ease the decision making process are vital.

6.3 Future work

Areas for investigation include:

6.3.1 Dataset

- Use other (larger) classification datasets of similar type/topic to validate findings
- Pre-process dataset further to test for robustness e.g. add noise (jitter the data), leave out some data
- Reduce dimensions to simplify the input attribute vector

6.3.2 METHODS

- Using variant algorithms and methods e.g. including second order methods, boosting or bagging
- Study the effect of initial weight settings
- Varying parameters tested to include e.g. momentum/regularization to assist in finding optimal solutions
- Vary the machine learning parameters for a 'fairer' comparison
- Further analyze the merits of oversampling (methods, domains)

ACKNOWLEDGEMENTS

The authors would like to acknowledge the work of S. Roland [22], K. Jeschkies [23] and E. Chen [24] for the Python algorithms adopted in this work.

REFERENCES

[1] Moro, S., Cortez, P. and Rita, P. 2014. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31.

[2] Moro, S., Laureano, R. and Cortez, P. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. 2011. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, EUROSIS.

[3] Chawla, V., Bowyer, K., Hall, L.. and Kegelmeyer, W. 2002. "SMOTE: synthetic minority over-sampling technique." Journal of Artificial Intelligence Research 16.1: pp.321-357.

[4] http://kumaranpm.blogspot.co.uk/2015/03/impact-of-target-class-proportions-on.html [Accessed 23/03/15].

[5] Hinton, G. & Salakhutdinov, R. 2006, "Reducing the Dimensionality of Data with Neural Networks", Science, vol. 313, no. 5786, pp. 504-507.

[6] Hinton, G. 2010. A practical guide to training restricted Boltzmann machines. Momentum, 9(1), 926.

[7] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.

[8] Vanhoucke, V., Senior, A., & Mao, M. 2011. Improving the speed of neural networks on CPUs. In Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop.

[9] http://www.iro.umontreal.ca/~bengioy/dlbook/ [Accessed 23/03/15].

[10] Nielsen,M. 2015. "Neural Networks and Deep Learning", Determination Press. http://neuralnetworksanddeeplearning.com/. [Accessed 24/03/15].

[11] Bengio, Y. 2012. Practical recommendations for gradient-based training of deep architectures. In Neural Networks: Tricks of the Trade, pp. 437-478. Springer: Berlin, Heidelberg.

[12] Giles, R. 2001. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference (Vol. 13, p. 402). MIT Press.

[13] Hinton, G. 2002. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8), 1771-1800.

[14] Lecun, Y. 1988. Generalization and network design strategies, Proceedings of the International Conference Connectionism in Perspective, University of Zurich, 10-13 October 1988.

[15] Haykin, S. 1999. Neural networks: a comprehensive foundation, Prentice Hall. Upper Saddle River, NJ.

[16] Haykin, S. 2009. Neural networks and learning machines, 3rd Ed. Prentice Hall. Upper Saddle River, NJ.

[17] Bishop, C. 2006. Pattern recognition and machine learning. Springer: New York.

[18] Mitchell, T. 1997. Machine learning, McGraw-Hill. New York: London.

[19] Murphy, K. 2012. Machine learning: a probabilistic perspective. MIT Press: London.

[20] Tan, P., Steinbach, M. & Kumar, V. 2013. Introduction to data mining, Addison-Wesley.Boston:London.

[21] Witten, H., Frank, E. & Hall, M. 2011. Data mining: practical machine learning tools and techniques, Morgan Kaufmann. Amsterdam: London.

[22] http://roliz.ro/2013/04/18/neural-networks-in-python/ [Accessed: 17/3/2015]

[23] https://github.com/blackblab/nyan/blob/master/shared_modules/smote/py [Accessed 17/3/2015]

[24] https://github.com/echen/restricted-boltzmann-machines/blob/master/rbm.py [Accessed: 17/3/2015]